**IN THE CLAIMS:**

The following listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended) A computer graphics system for generating pixels from a distributed convolution of rendered samples comprising:
   a plurality of sample managers connected in series; and
   a set of partial sums buses, wherein each partial sums bus connects one of the sample managers of the series to the next sample manager in the series;
   wherein each sample manager is operable to calculate partial sums for a corresponding portion of the rendered samples located within a convolution kernel corresponding to a pixel location, wherein the partial sums comprise 1) a sum of weights determined for locations of the rendered samples in the portion of rendered samples and 2) a sum of weighted sample values for the portion of rendered samples,
   wherein each of the second through the last sample manager in the series is operable to add the partial sums calculated for its corresponding portion of the rendered samples to any previously accumulated partial sums received from the prior sample manager in the series, and if not the last sample manager in the series, output new accumulated partial sums to the next sample manager in the series.

2. (Previously Presented) The system of claim 1, wherein the last sample manager in the series calculates pixel values from the final accumulated partial sums.

3. (Currently Amended) The system of claim 1, wherein for each sample manager the corresponding portion of rendered samples resides in a sub-set of screen space and the sub-set is interleaved across screen space.

4. (Previously Presented) The system of claim 3, comprising 16 sample managers, wherein each sample manager addresses one sample bin in a 4 by 4 array of sample bins and a corresponding sample bin in each repetition of the 4 by 4 array to span screen space.

5. (Previously Presented) The system of claim 1, further comprising a plurality of groups of sample managers and a set of partial sums buses interconnecting the groups, wherein each group's sample managers are connected in series.

6. (Currently Amended) The system of claim 5, wherein for each sample manager the corresponding portion of rendered samples resides in a corresponding sub-set of screen space and the sub-sets are interleaved across screen space, wherein for a system of 4 groups of 4 sample managers in a series, each sample manager within a group addresses one sample bin in a 2 by 2 array of sample bins that is repeated across a 16 by 16 array of sample bins, and wherein four permutations of each of the four different 16 by 16 arrays (one for each group) are combined to form a 64 by 64 array of sample bins that is repeated across screen space.

7. (Original) The system of claim 5, wherein a last sample manager in each group calculates pixel values from the final accumulated partial sums.

8. (Currently Amended) A system for distributed filtering of samples within a convolution kernel to calculate values for a corresponding pixel comprising:
a series of means for calculating partial sums,
wherein each member of the series calculates partial sums for a corresponding portion of the samples within the convolution kernel for the pixel,
wherein the partial sums calculated by each member of the series comprise 1) a sum of weights determined for the sample locations in the corresponding portion of samples and 2) a sum of weighted sample values for the corresponding portion of samples,

wherein each member of the series adds the calculated partial sums to corresponding accumulated partial sums and outputs the new accumulated partial sums,

and wherein a last member of the series calculates pixel values from the final accumulated partial sums; and

means for passing accumulated partial sums from one member of the series to the next;

a plurality of means for storing sample values, wherein each means for storing sample values is dedicated to a different member of the series; and

a plurality of means for rendering samples, wherein each sample generated is stored in one of the means for storing sample values.

9. (Original) The system of claim 8, wherein a sample comprises parameter values for color and transparency, and wherein partial sums comprise partial sums for each of the parameter values.

10. (Original) The system of claim 8, wherein each means for calculating partial sums is assigned one or more sample bins from an interleaved array of sample bins and the interleaved array of sample bins is repeated across screen space.

11. - 12. (Cancelled).

13. (Previously Presented) The system of claim 8, further comprising a means for converting the pixel values to video output signals.

14. (Previously Presented) A system for distributed filtering of samples comprising:

a series of N sample managers (k), wherein k is an integer with range 0 to N-1;

a partial sums bus connecting each sample manager (k) in the series of sample managers to the next sample manager (k+1);

wherein sample manager (k) is operable to:

receive accumulated partial sums from a prior sample manager (k-1), if k is greater than zero, wherein partial sums comprise partial sums for each sample parameter value,

calculate partial sums for a set of samples, wherein the set of samples are within a sub-set of screen space assigned to sample manager (k), and wherein the set of samples are located within a convolution kernel defined for a pixel,

add the partial sums to the accumulated partial sums, and

output the accumulated partial sums to sample manager (k+1), if k is less than N-1; and

wherein a designated sample manager is operable to calculate pixel values from the final accumulated partial sums.

15. (Original) The system of claim 14, further comprising a memory (k) dedicated to sample manager (k), wherein memory (k) stores sample data for samples in the sub-set of screen space assigned to the sample manager (k).

16. (Original) The system of claim 15, wherein each memory (k) comprises a plurality of memory units.

17. (Original) The system of claim 15, wherein sample manager (k) is operable to read the set of samples from the memory (k).

18. - 19. (Canceled).

20. (Previously Presented) The system of claim 14, wherein each pixel parameter value equals a corresponding final accumulated sum of weighted sample parameter values for each sample within the convolution kernel divided by an accumulated sum of weights for locations of each sample within the convolution kernel.

21. (Currently Amended) A computer graphics system for generating pixels from a distributed convolution of rendered samples ~~for a pixel~~ comprising:

a set of N memories, wherein each memory (k) stores sample data for rendered samples in a different sub-set (k) of screen space, wherein k is a non negative integer;

a sequence of N filter units, wherein each filter unit (k) is directly connected to a dedicated memory (k); and

a set of N-1 partial sums buses connecting each filter unit to the next filter unit in the sequence,

wherein each filter unit (k) is operable to:

receive accumulated partial sums from a prior filter unit (k-1), if k is greater than zero,

read a set of rendered samples from a memory (k), wherein the set of rendered samples are within sub-set (k) of screen space assigned to the memory (k), and wherein the set of rendered samples are located within a convolution kernel defined for the pixel,

calculate partial sums for the set of rendered samples,

add the partial sums to the accumulated partial sums, and

output the accumulated partial sums to the next filter unit in the sequence of filter units if k is less than N-1,

and wherein the last filter unit (N-1) in the sequence calculates pixel values from the final accumulated sums.

22. (Previously Presented) The system of claim 21, wherein N = 16 and the 16 different sub-sets of screen space are interleaved so that each filter unit addresses one sample bin in a 4 by 4 array of sample bins that is repeated across screen space.

23. (Original) A system for distributed convolution of samples for a pixel comprising:

a set of N memories, wherein each memory (k) stores sample data for samples located within sub-set (k) of screen space, wherein k is an integer ranging from 0 to N-1;

a sequence of N filter units arranged in M groups, wherein each filter unit (k) has a dedicated memory (k);

a first set of partial sums buses that connect each filter unit within a group in series;

a second set of partial sums buses that connect one group of filter units to another group of filter units;

wherein filter units within a group are operable to:

receive accumulated partial sums from a prior filter unit within the group or from another group of filter units,

read a set of samples from a corresponding memory (k), wherein the set of samples are within the sub-set (k) of screen space assigned to the memory (k), and wherein the set of samples are located within a convolution kernel defined for the pixel,

calculate partial sums for the set of samples,

add the partial sums to the accumulated partial sums, and

output the accumulated partial sums to the next filter unit within the group of filter units, or to another group of filter units,

and wherein a last filter unit within a group calculates pixel values from the final accumulated sums, after all samples within the convolution kernel have been processed.

24. (Original) The system of claim 23, wherein N = 16, M = 4, and the 16 different sub-sets of screen space are interleaved, wherein each filter unit within a group addresses one sample bin in a 2 by 2 array of sample bins that is repeated across a 16 by 16 array of sample bins, and wherein four permutations of each of the four different 16 by 16 arrays (one for each group) are combined to form a 64 by 64 array of sample bins that is repeated across screen space.

25. (Currently Amended) A method for distributed filtering of rendered samples comprising:

calculating first partial sums in a first filter unit for a first set of rendered samples, wherein the first set of rendered samples is a portion of the rendered samples

located within a convolution kernel defined for a pixel location, and wherein the first set of rendered samples are within a region of screen space assigned to the first filter unit; and

sending the first partial sums to a sequence of additional filter units, wherein each of the additional filter units:

receives accumulated partial sums from the previous filter unit,

calculates new partial sums for a corresponding set of rendered samples located within the convolution kernel and within a corresponding region of screen space assigned to the filter unit,

adds the new partial sums to the accumulated partial sums, and

sends the new accumulated partial sums to the next filter unit in the sequence of filter units; and

wherein a last filter unit in the sequence of filter units:

receives accumulated partial sums from the previous filter unit in the sequence,

calculates new partial sums for a corresponding portion of rendered samples, and

adds the new partial sums to the accumulated partial sums to complete final accumulated partial sums.

26. (Original) The method of claim 25, wherein a convolution kernel defined for a pixel is a region in screen space within a defined boundary and centered on the pixel location in screen space.

27. (Currently Amended) The method of claim 25, wherein partial sums comprise 1) a sum of weighted sample values for the set of rendered samples and 2) a sum of the weights determined for the locations of each rendered sample, wherein each weighted sample value is a product of one of the sample values and a weight determined for the corresponding location.

28. (Original) The method of claim 27, wherein a weight for the location of each sample value is determined by a weight function selected from a set of functions comprising a box filter, pyramid filter, circular filter, cone filter, Gaussian filter, and sinc filter.

29. (Canceled).

30. (Original) The method of claim 25, wherein the last filter unit of the sequence of filter units calculates pixel values from the final accumulated sums.

31. (Canceled).

32. (Currently Amended) The method of claim 25, wherein each set of rendered samples is within a screen space region (k) assigned to the corresponding filter unit (k), and wherein the set of rendered samples are read from a memory (k) dedicated to the filter unit (k).

33. (Currently Amended) The method of claim 25, wherein for each filter unit (k) the corresponding set of rendered samples resides in a sub-set (k) of screen space and the sub-sets are interleaved across screen space.

34. (Original) The method of claim 33, wherein for a system of 16 filter units, each filter unit (k) addresses one sample bin in a 4 by 4 array of sample bins that is repeated across screen space.

35. (Previously Presented) The method of claim 25, wherein the filter units are sub-divided into a plurality of groups of filter units and a plurality of partial sums buses interconnect the groups, and wherein each group is a series of filter units.

36. (Original) The method of claim 35, wherein a last filter unit in each group calculates pixel values from the final accumulated partial sums corresponding to the pixel.

37. (Currently Amended) The method of claim 35, wherein for each filter unit (k) the set of rendered samples resides in a sub-set (k) of screen space and the sub-sets are interleaved across screen space.

38. (Previously Presented) The method of claim 37, wherein for a system of 4 groups of 4 filter units in a series, each filter unit within a group addresses a sample bin in a 2 by 2 array of sample bins that is repeated across a 16 by 16 array, and wherein four permutations of each of the four different 16 by 16 arrays (one for each group) are combined to form a 64 by 64 array of sample bins that is repeated across screen space.

39. (Previously Presented) A system for distributed filtering of samples comprising:
a series of N sample managers (k), wherein k is an integer with range 0 to N-1;
a partial sums bus connecting each sample manager (k) in the series of sample managers to the next sample manager (k+1);
wherein sample manager (k) is operable to:
receive accumulated partial sums from a prior sample manager (k-1), if k is greater than zero,
calculate partial sums for a set of samples, wherein the set of samples are within a sub-set of screen space assigned to sample manager (k), and wherein the set of samples are located within a convolution kernel defined for a pixel,
add the partial sums to the accumulated partial sums, and
output the accumulated partial sums to sample manager (k+1), if k is less than N-1; and
a memory (k) dedicated to sample manager (k), wherein memory (k) stores sample data for samples in the sub-set of screen space assigned to the sample manager (k), and wherein each memory (k) comprises a plurality of separate memory units.

40. (Previously Presented) A method for distributed filtering of samples comprising:

calculating first partial sums in a first filter unit for a first set of samples, wherein partial sums comprise a sum of weighted sample values for the set of samples and a sum of the weights determined for the locations of each sample, wherein a weighted sample value is a product of the sample value and the determined weight for the location of the sample, wherein sample values comprise color values and transparency, wherein the first set of samples is a portion of the samples located within a convolution kernel defined for a pixel location, and wherein the first set of samples are within a region of screen space assigned to the first filter unit; and

sending the first partial sums to a sequence of additional filter units, wherein each of the additional filter units:

receives accumulated partial sums from the previous filter unit,

calculates new partial sums for a corresponding set of samples located within the convolution kernel and within a corresponding region of screen space assigned to the filter unit,

adds the new partial sums to the accumulated partial sums, and

sends the new accumulated partial sums to the next filter unit in the sequence of filter units; and

wherein a last filter unit in the sequence of filter units:

receives accumulated partial sums from the previous filter unit in the sequence,

calculates new partial sums for a corresponding portion of samples, and

adds the new partial sums to the accumulated partial sums to complete final accumulated partial sums.

41. (Previously Presented) A method for distributed filtering of samples comprising:

calculating first partial sums in a first filter unit for a first set of samples, wherein the first set of samples is a portion of the samples located within a convolution kernel defined for a pixel location, and wherein the first set of samples are within a region of screen space assigned to the first filter unit; and

sending the first partial sums to a sequence of additional filter units, wherein each of the additional filter units:

receives accumulated partial sums from the previous filter unit,

calculates new partial sums for a corresponding set of samples located within the convolution kernel and within a corresponding region of screen space assigned to the filter unit,

adds the new partial sums to the accumulated partial sums, and

sends the new accumulated partial sums to the next filter unit in the sequence of filter units; and

wherein a last filter unit in the sequence of filter units:

receives accumulated partial sums from the previous filter unit in the sequence,

calculates new partial sums for a corresponding portion of samples,

calculates final sums by adding the new partial sums to the accumulated partial sums, and

calculates pixel values from the final sums, by multiplying a final sum of weighted sample values times a reciprocal of a final sum of the weights for each parameter value comprising one or more of color values and transparency.